

```

// Computer Program Listing Appendix Under 37 CFR 1.52(e)
// DCCode (Text).txt
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.
Code to save and restore the custom/extra information to/from the Data Consolidator.
//
// DCEExtendABeanXML.java
//
package com.ffusion.dataconsolidator.adapter;
import java.sql.*;
import java.util.*;
import com.ffusion.util.db.DBUtil;
class DCEExtendABeanXML
{
    // SQL queries
    private static final String SQL_ADD_XML = "INSERT INTO DC_ExtendABeanXML( DCEExtendABeanID,
XMLSegmentNumber, XMLSegment ) VALUES( ?, ?, ?)";
    private static final String SQL_DELETE_XML = "DELETE FROM DC_ExtendABeanXML WHERE
DCEExtendABeanID=?";
    private static final String SQL_GET_XML = "SELECT XMLSegment FROM DC_ExtendABeanXML WHERE
DCEExtendABeanID=? ORDER BY XMLSegmentNumber ASC";
    /**
     Add new ExtendABean XML to the DC_ExtendABeanXML table
     @param XML: ExtendABean XML to be added
     @return: the DCEExtendABeanID of the new row
     */
    protected static long addExtendABeanXML( Connection connection, String XML, HashMap extra ) throws
DCEException
    {
        if( XML == null || (XML.length() == 0) ) {
            return 0;
        }
        PreparedStatement stmt = null;
        try {
            long id = DCRecordCounter.getNextIndex( connection, DCRecordCounter.TYPE_EXTENDABEAN,
DCRecordCounter.EXTENDABEAN_ID, DCRecordCounter.EXTENDABEAN_INDEX, extra );
            // get the prepared statement
            stmt = DCAdapter.getStatement( connection, SQL_ADD_XML );
            int segmentNumber = 1;
            String segment = null;
            boolean notDone = true;
            while( notDone ) {
                if( XML.length() > 2000 ) {
                    segment = XML.substring( 0, 2000 );
                    XML = XML.substring( 2000, XML.length() );
                } else {
                    segment = XML;
                    notDone = false;
                }
            }
            stmt.setLong( 1, id );
            stmt.setInt( 2, segmentNumber );

```

```

stmt.setString( 3, segment );
DBUtil.executeUpdate( stmt, SQL_ADD_XML );
segmentNumber++;
}
return id;
} catch ( Exception e ) {
    throw new DCException( e );
} finally {
    stmt = null;
}
}
/**

```

Get the XML from the database, put it back together in order, then return a String containing the XML

@param id: DCExtendABeanID of XML to retrieve

@return: String containing the ExtendABean XML

```

*/
protected static String getExtendABeanXML( Connection connection, long id ) throws DCException
{
    // an id of 0 means that there is no ExtendABeanXML
    if( id == 0 ) {
        return null;
    }
    PreparedStatement stmt = null;
    ResultSet rs = null;
    try {
        stmt = DCAdapter.getStatement( connection, SQL_GET_XML );
        stmt.setLong( 1, id );
        rs = DBUtil.executeQuery( stmt, SQL_GET_XML );
        StringBuffer sb = new StringBuffer();
        while( rs.next() ) {
            // put the xml back together again in the correct order
            sb.append( rs.getString( 1 ) );
        }
        return sb.toString();
    } catch ( Exception e ) {
        throw new DCException( e );
    } finally {
        stmt = null;
        if( rs != null ) {
            try {
                rs.close();
            } catch( Exception ex ) {
            }
        }
    }
}

```

Code to assign sequence numbers for transactions

```

//
// DCRecordCounter.java

```

```

//
package com.ffusion.dataconsolidator.adapter;
import java.sql.*;
import com.ffusion.util.MapUtil;
import java.util.*;
import com.ffusion.util.db.DBUtil;
import com.ffusion.dataconsolidator.DCConstants;
class DCRecordCounter
{
    // Object Types
    protected static final int TYPE_ACCOUNT = 1;
    protected static final int TYPE_LOCKBOX = 2;
    protected static final int TYPE_EXTENDABEAN = 3;
    // Object ID
    protected static final int EXTENDABEAN_ID = 0;
    // Counter name constants
    protected static final String ACCT_TRANSACTION_INDEX = "ACCT_TRANSACTION_INDEX";
    protected static final String LOCKBOX_CREDITITEM_INDEX = "LOCKBOX_CREDITITEM_INDEX";
    protected static final String LOCKBOX_TRANSACTION_INDEX = "LOCKBOX_TRANSACTION_INDEX";
    protected static final String DISBURSEMENT_TRANSACTION_INDEX =
"DISBURSEMENT_TRANSACTION_INDEX";
    protected static final String EXTENDABEAN_INDEX = "DCExtendABeanIndex";
    // SQL queries
    private static final String SQL_INCREMENT_INDEX = "UPDATE DC_RecordCounter SET NextIndex=NextIndex+1
WHERE DCOBJECTType=? AND DCOBJECTID=? AND CounterName=? AND DataClassification=? ";
    private static final String SQL_GET_INDEX = "SELECT NextIndex FROM DC_RecordCounter WHERE
DCOBJECTType=? AND DCOBJECTID=? AND CounterName=? AND DataClassification=? ";
    // Get the index for the given counter
    protected static long getIndex( Connection connection, int type, int objectID, String counterName, HashMap extra )
throws DCException
    {
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            String dataClassification = null;
            // the record counter will always have a data classification of P for the extendabean index
            if ( type == DCRecordCounter.TYPE_EXTENDABEAN && objectID == DCRecordCounter.EXTENDABEAN_ID &&
counterName.equals( DCRecordCounter.EXTENDABEAN_INDEX ) ) {
                dataClassification = DCConstants.DATA_CLASSIFICATION_PREVIOUSDAY;
            } else {
                dataClassification = MapUtil.getStringValue( extra, DCConstants.DATA_CLASSIFICATION,
                    DCConstants.DATA_CLASSIFICATION_PREVIOUSDAY );
            }
            stmt = DCAdapter.getStatement( connection, SQL_GET_INDEX );
            stmt.setInt( 1, type );
            stmt.setInt( 2, objectID );
            stmt.setString( 3, counterName );
            stmt.setString( 4, dataClassification );
            rs = DBUtil.executeQuery( stmt, SQL_GET_INDEX );
            if( rs.next() ) {

```

```

    long index = rs.getLong( 1 );
        return index;
    } else {
        throw new DCException( "Error occurred while retrieving index." );
    }
} catch ( Exception e ) {
    throw new DCException( e );
} finally {
    stmt = null;
    DBUtil.closeResultSet( rs );
}
}

// increment the specified index by one
protected static void incrementIndex( Connection connection, int type, int objectID, String counterName, HashMap
extra ) throws DCException
{
    PreparedStatement stmt = null;
    String dataClassification = null;
    // the record counter will always have a data classification of P for the extendabean index
    if ( type == DCRecordCounter.TYPE_EXTENDABEAN && objectID == DCRecordCounter.EXTENDABEAN_ID &&
        counterName.equals( DCRecordCounter.EXTENDABEAN_INDEX ) ) {
        dataClassification = DCConstants.DATA_CLASSIFICATION_PREVIOUSDAY;
    } else {
        dataClassification = MapUtil.getStringValue( extra, DCConstants.DATA_CLASSIFICATION,
            DCConstants.DATA_CLASSIFICATION_PREVIOUSDAY );
    }
    try {
        stmt = DCAdapter.getStatement( connection, SQL_INCREMENT_INDEX );
        stmt.setInt( 1, type );
        stmt.setInt( 2, objectID );
        stmt.setString( 3, counterName );
        stmt.setString( 4, dataClassification );
        DBUtil.executeUpdate( stmt, SQL_INCREMENT_INDEX );
    } catch ( Exception e ) {
        throw new DCException( e );
    } finally {
        stmt = null;
    }
}

// the current index in the table refers to the last index
// so increment the current index, and return the new value
protected static long getNextIndex( Connection connection, int type, int objectID, String counterName, HashMap
extra ) throws DCException
{
    PreparedStatement stmt1 = null;
    PreparedStatement stmt2 = null;
    ResultSet rs = null;
    long index = 0;
    String dataClassification = null;
    // the record counter will always have a data classification of P for the extendabean index

```

```

if ( type == DCRecordCounter.TYPE_EXTENDABEAN && counterName.equals(
DCRecordCounter.EXTENDABEAN_INDEX ) ) {
    dataClassification = DCConstants.DATA_CLASSIFICATION_PREVIOUSDAY;
} else {
    dataClassification = MapUtil.getStringValue( extra, DCConstants.DATA_CLASSIFICATION,
        DCConstants.DATA_CLASSIFICATION_PREVIOUSDAY );
}

    try {
        // get the prepared statements
        stmt1 = DCAdapter.getStatement( connection, SQL_INCREMENT_INDEX );
        stmt2 = DCAdapter.getStatement( connection, SQL_GET_INDEX );
        stmt1.setInt( 1, type );
        stmt1.setInt( 2, objectID );
        stmt1.setString( 3, counterName );
        stmt1.setString( 4, dataClassification );
        DBUtil.executeUpdate( stmt1, SQL_INCREMENT_INDEX );
        stmt2.setInt( 1, type );
        stmt2.setInt( 2, objectID );
        stmt2.setString( 3, counterName );
        stmt2.setString( 4, dataClassification );
        rs = DBUtil.executeQuery( stmt2, SQL_GET_INDEX );
        if( rs.next() ) {
            index = rs.getLong( 1 );
        } else {
            throw new DCException( "Error occurred while retrieving index." );
        }
        return index;
    } catch ( Exception e ) {
        throw new DCException( e );
    } finally {
        stmt1 = null;
        stmt2 = null;
        DBUtil.closeResultSet( rs );
    }
}

```

Code to get the next page of data (this example retrieves account transactions)

```

private static final String SQL_GET_NEXTTRANS = "SELECT b.DCTransactionIndex, b.DataDate, b.TransID, " +
    "b.TransTypeID, b.TransCategoryID, b.TransTrackingID, b.Description, b.Memo, b.ReferenceNumber, b.Amount, "
+
    "b.RunningBalance, b.ImmedAvailAmount, b.OneDayAvailAmount, b.MoreOneDayAvailAm, b.ValueDateTime,
b.BankRefNum, " +
    "b.CustRefNum, b.PostingDate, b.DueDate, b.FixedDepRate, b.PayeePayor, b.PayorNum, b.OrigUser, b.PONum,
b.ExtendABeanXMLID, b.InstNumber, b.InstBankName, b.TransSubTypeID, TransDate " +
    "FROM DC_Transactions b, DC_Account a WHERE b.DCAccountID=a.DCAccountID AND a.AccountID=? AND
a.BankID=? " +
    "AND b.DCTransactionIndex >= ? AND b.DCTransactionIndex<=? AND DataClassification=? ";
/*

```

Retrieves the next batch of PAGESIZE transactions for the specified account

@param account: the account for which we want to retrieve the transactions

@param nextIndex: the next index of information to retrieve

```

@return: a list of Transaction beans containing the transactions (at most PAGE_SIZE of them)
*/
protected static com.ffusion.beans.banking.Transactions getNextTransactions(
com.ffusion.beans.accounts.Account account,
    long nextIndex,
    HashMap extra )
    throws DCException
{
PreparedStatement stmt = null;
Transactions transactions = null;
Connection connection = null;
ResultSet rs = null;
StringBuffer sb = new StringBuffer();
int pageSize = MapUtil.getIntValue( extra, DCConstants.PAGE_SIZE, DCAdapter.PAGESIZE );
String dataClassification = MapUtil.getStringValue( extra, DCConstants.DATA_CLASSIFICATION,
    DCConstants.DATA_CLASSIFICATION_PREVIOUSDAY );
try {
    connection = DCAdapter.getDBConnection();
    sb.append( SQL_GET_NEXTTRANS );
    DCUtil.appendNullWhereClause( sb, ROUTINGNUM, account.getRoutingNum() );
    sb.append( ORDERBY_CLAUSE );
    stmt = DCAdapter.getStatement( connection, sb.toString() );
    stmt.setString( 1, account.getID() );
    stmt.setString( 2, account.getBankID() );
    stmt.setLong( 3, nextIndex );
    stmt.setLong( 4, nextIndex + pageSize - 1 );
    stmt.setString( 5, dataClassification );
    if( account.getRoutingNum() != null ) {
stmt.setString( 6, account.getRoutingNum() );
    }
    rs = DBUtil.executeQuery( stmt, sb.toString() );
    transactions = new Transactions( );
    Transaction trans = null;
    while( rs.next() ) {
trans = transactions.create();
loadTransaction( trans, rs );
    }
    return transactions;
} catch ( Exception e ) {
    throw new DCException( e );
} finally {
    stmt = null;
    DBUtil.closeResultSet( rs );
    if( connection != null ) {
DCAdapter.releaseDBConnection( connection );
    }
}
}
}
// dcdb2.txt
// Copyright (c) 2004. Sybase, Inc. All Rights Reserved.

```

```

-- =====
-- This table stores a list of accounts. An account is identified by
-- AccountID and BankID, or by DCAccountID.
-- =====

CREATE TABLE DC_Account (
  DCAccountID INTEGER NOT NULL,
  BankID VARCHAR(40) NOT NULL,
  AccountID VARCHAR(40) NOT NULL,
  CurrencyCode CHAR(3) NOT NULL,
  RoutingNumber VARCHAR(100),
  Extra VARCHAR(1024)
);
ALTER TABLE DC_Account ADD PRIMARY KEY (DCAccountID);
CREATE UNIQUE INDEX DC_Account1 ON DC_Account (AccountID, BankID, RoutingNumber );
CREATE SEQUENCE DCAccountID_seq start with 1 increment by 1 nomaxvalue nocycle;
-- =====

-- This table stores status information about accounts. Status information
-- can be available for many days. Each record indicates the status of an
-- account on the date specified in the DataDate column. The DataSource
-- column indicates the source from which the data came (possible choices
-- are BAI file, SWIFT connection, or live from a back-end).
-- =====

CREATE TABLE DC_AccountHistory (
  DCAccountID INTEGER NOT NULL,
  DataDate TIMESTAMP NOT NULL,
  DataSource INTEGER NOT NULL,
  OpeningLedger NUMERIC(31,3),
  AvgOpenLedgerMTD NUMERIC(31,3),
  AvgOpenLedgerYTD NUMERIC(31,3),
  ClosingLedger NUMERIC(31,3),
  AvgCloseLedgerMTD NUMERIC(31,3),
  AvgMonth NUMERIC(31,3),
  AggBalanceAdjust NUMERIC(31,3),
  AvCloseLedgYTDPvM NUMERIC(31,3),
  AvgCloseLedgerYTD NUMERIC(31,3),
  CurrentLedger NUMERIC(31,3),
  ACHNetPosition NUMERIC(31,3),
  OpAvaiSamDayACHDTC NUMERIC(31,3),
  OpeningAvailable NUMERIC(31,3),
  AvgOpenAvailMTD NUMERIC(31,3),
  AvgOpenAvailYTD NUMERIC(31,3),
  AvgAvailPrevMonth NUMERIC(31,3),
  DisbOpenAvailBal NUMERIC(31,3),
  ClosingAvail NUMERIC(31,3),
  AvgCloseAvailMTD NUMERIC(31,3),
  AvCloseAvailPreM NUMERIC(31,3),
  AvCloseAvailYTDPvM NUMERIC(31,3),
  AvCloseAvailYTD NUMERIC(31,3),
  LoanBalance NUMERIC(31,3),
  TotalInvestPosn NUMERIC(31,3),

```

```

CurrAvailCRSSupr NUMERIC(31,3),
CurrentAvail NUMERIC(31,3),
AvgCurrAvailMTD NUMERIC(31,3),
AvgCurrAvailYTD NUMERIC(31,3),
TotalFloat NUMERIC(31,3),
TargetBalance NUMERIC(31,3),
AdjBalance NUMERIC(31,3),
AdjBalanceMTD NUMERIC(31,3),
AdjBalanceYTD NUMERIC(31,3),
ZeroDayFloat NUMERIC(31,3),
OneDayFloat NUMERIC(31,3),
FloatAdj NUMERIC(31,3),
TwoMoreDayFloat NUMERIC(31,3),
ThreeMoreDayFloat NUMERIC(31,3),
AdjToBalances NUMERIC(31,3),
AvgAdjToBalMTD NUMERIC(31,3),
AvgAdjToBalYTD NUMERIC(31,3),
FourDayFloat NUMERIC(31,3),
FiveDayFloat NUMERIC(31,3),
SixDayFloat NUMERIC(31,3),
AvgOneDayFloatMTD NUMERIC(31,3),
AvgOneDayFloatYTD NUMERIC(31,3),
AvgTwoDayFloatMTD NUMERIC(31,3),
AvgTwoDayFloatYTD NUMERIC(31,3),
TransferCalc NUMERIC(31,3),
TargBalDeficiency NUMERIC(31,3),
TotalFundingReq NUMERIC(31,3),
TotalChecksPaid NUMERIC(31,3),
GrandTotCredMinDeb NUMERIC(31,3),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_AccountHistory ADD PRIMARY KEY (DCAccountID, DataDate, DataClassification);
ALTER TABLE DC_AccountHistory ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;
-- =====
-- This table stores summary information about accounts.
-- =====
CREATE TABLE DC_AccountSummary (
DCAccountID INTEGER NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
TotalCredits NUMERIC(31,3),
TotalCreditsMTD NUMERIC(31,3),
CreditsNotDetailed NUMERIC(31,3),
DepositsSubjFloat NUMERIC(31,3),

```



```

TotalAdjCreditsYTD NUMERIC(31,3),
TotalLockboxDeposits NUMERIC(31,3),
TotalDebits NUMERIC(31,3),
TotalDebitsMTD NUMERIC(31,3),
TodayTotalDebits NUMERIC(31,3),
TotalDebitLessWire NUMERIC(31,3),
TotalAdjDebitsYTD NUMERIC(31,3),
TotalDebitsExRetn NUMERIC(31,3),
ImmedAvailAmount NUMERIC(31,3),
OneDayAvailAmount NUMERIC(31,3),
MoreOneDayAvailAm NUMERIC(31,3),
ValueDateTime TIMESTAMP,
AvailableOverdraft NUMERIC(31,3),
RestrictedCash NUMERIC(31,3),
AccruedInterest NUMERIC(31,3),
AccruedDividend NUMERIC(31,3),
TotalOverdraftAmt NUMERIC(31,3),
NxtOvrDrftPmtDate TIMESTAMP,
InterestRate FLOAT,
BookValue NUMERIC(31,3),
MarketValue NUMERIC(31,3),
OpeningLedger NUMERIC(31,3),
ClosingLedger NUMERIC(31,3),
CurrAvailBal NUMERIC(31,3),
LedgerBal NUMERIC(31,3),
OneDayFloat NUMERIC(31,3),
TwoDayFloat NUMERIC(31,3),
TotalFloat NUMERIC(31,3),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_AccountSummary ADD PRIMARY KEY (DCAccountID, DataDate, DataClassification);
ALTER TABLE DC_AccountSummary ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;
-- =====
-- This table stores extended summary information about accounts. The most
-- common summary information is in the DC_AccountSummary table.
-- =====
CREATE TABLE DC_ExtAcctSummary (
DCAccountID INTEGER NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
SummaryType INTEGER NOT NULL,
Amount NUMERIC(31,3),
ImmedAvailAmount NUMERIC(31,3),
OneDayAvailAmount NUMERIC(31,3),

```

```

MoreOneDayAvailAm NUMERIC(31,3),
ValueDateTime TIMESTAMP,
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_ExtAcctSummary ADD PRIMARY KEY (DCAccountID, DataDate, SummaryType,
DataClassification);
ALTER TABLE DC_ExtAcctSummary ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;

-- =====
-- This table stores information about individual transactions within an
-- account. This table can get quite large.
-- =====

CREATE TABLE DC_Transactions (
DCAccountID INTEGER NOT NULL,
DCTransactionIndex BIGINT NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
TransID VARCHAR(40),
TransTypeID INTEGER,
TransCategoryID INTEGER,
TransTrackingID VARCHAR(40),
TransSubTypeID INTEGER,
Description VARCHAR(1024),
Memo VARCHAR(255),
ReferenceNumber VARCHAR(40),
Amount NUMERIC(31,3),
RunningBalance NUMERIC(31,3),
ImmedAvailAmount NUMERIC(31,3),
OneDayAvailAmount NUMERIC(31,3),
MoreOneDayAvailAm NUMERIC(31,3),
ValueDateTime TIMESTAMP,
BankRefNum VARCHAR(40),
CustRefNum VARCHAR(40),
PostingDate TIMESTAMP,
DueDate TIMESTAMP,
FixedDepRate FLOAT,
PayeePayor VARCHAR(40),
PayorNum VARCHAR(40),
OrigUser VARCHAR(40),
PONum VARCHAR(40),
InstNumber VARCHAR(40),
InstBankName VARCHAR(80),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),

```

```

DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
TransDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_Transactions ADD PRIMARY KEY (DCAccountID, DCTransactionIndex, DataClassification);
ALTER TABLE DC_Transactions ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;

-- =====
-- This table holds data specific to Credit Card Account
-- =====

CREATE TABLE DC_CCAcctSummary (
DCAccountID INTEGER NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
AvailableCredit NUMERIC(31,3),
AmountDue NUMERIC(31,3),
InterestRate FLOAT,
DueDate TIMESTAMP,
CardHolderName VARCHAR(1024),
CardExpDate TIMESTAMP,
CreditLimit NUMERIC(31,3),
LastPaymentAmt NUMERIC(31,3),
NextPaymentMinAmt NUMERIC(31,3),
NextPaymentDue TIMESTAMP,
LastPaymentDate TIMESTAMP,
ValueDate TIMESTAMP,
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_CCAcctSummary ADD PRIMARY KEY (DCAccountID, DataDate, DataClassification);
ALTER TABLE DC_CCAcctSummary ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;

-- =====
-- This table holds data specific to Loan Account
-- =====

CREATE TABLE DC_LoanAcctSummary (
DCAccountID INTEGER NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
AvailableCredit NUMERIC(31,3),
AmountDue NUMERIC(31,3),
InterestRate FLOAT,
DueDate TIMESTAMP,
MaturityDate TIMESTAMP,
AccruedInterest NUMERIC(31,3),

```

```

OpeningBalance NUMERIC(31,3),
CollateralDesc VARCHAR(1024),
PrinciplePastDue NUMERIC(31,3),
InterestPastDue NUMERIC(31,3),
LateFees NUMERIC(31,3),
NextPrincipleAmt NUMERIC(31,3),
NextInterestAmt NUMERIC(31,3),
ValueDate TIMESTAMP,
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_LoanAcctSummary ADD PRIMARY KEY (DCAccountID, DataDate, DataClassification);
ALTER TABLE DC_LoanAcctSummary ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON
DELETE CASCADE;

-- =====
-- This table holds data specific to Loan Account
-- =====

CREATE TABLE DC_FixDeplnstrment (
DCAccountID INTEGER NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
InstNumber VARCHAR(40) NOT NULL,
InstBankName VARCHAR(80) NOT NULL,
Currency CHAR(3),
PrincipalAmount NUMERIC(31,3),
AccruedInterest NUMERIC(31,3),
InterestAtMaturity NUMERIC(31,3),
ProceedsAtMaturity NUMERIC(31,3),
ValueDate TIMESTAMP,
MaturityDate TIMESTAMP,
RestrictedAmount NUMERIC(31,3),
NumberOfRollovers INTEGER,
DaysInTerm INTEGER,
InterestRate FLOAT,
SettleInstrType INTEGER,
TargetAcctNumber VARCHAR(40),
TargetRoutNumber VARCHAR(100),
StmtMail1Street1 VARCHAR(40),
StmtMail1Street2 VARCHAR(40),
StmtMail1City VARCHAR(20),
StmtMail1State VARCHAR(2),
StmtMail1Country VARCHAR(30),
StmtMail1Zip VARCHAR(11),
StmtMail2Street1 VARCHAR(40),
StmtMail2Street2 VARCHAR(40),
StmtMail2City VARCHAR(20),

```

```

StmtMail2State VARCHAR(2),
StmtMail2Country VARCHAR(30),
StmtMail2Zip VARCHAR(11),
StmtMail3Street1 VARCHAR(40),
StmtMail3Street2 VARCHAR(40),
StmtMail3City VARCHAR(20),
StmtMail3State VARCHAR(2),
StmtMail3Country VARCHAR(30),
StmtMail3Zip VARCHAR(11),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_FixDepInstrment ADD PRIMARY KEY (DCAccountID, DataDate, InstNumber, InstBankName,
DataClassification);
ALTER TABLE DC_FixDepInstrment ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;
-- =====
-- Lockbox Support
-- =====
-- =====
-- This table stores a list of lockboxes. There can be multiple
-- lockboxes for an account.
-- =====
CREATE TABLE DC_Lockbox (
  DCLockboxID INTEGER NOT NULL,
  DCAccountID INTEGER NOT NULL,
  LockboxNumber VARCHAR(40) NOT NULL,
  Extra VARCHAR(1024)
);
ALTER TABLE DC_Lockbox ADD PRIMARY KEY (DCLockboxID);
ALTER TABLE DC_Lockbox ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;
CREATE UNIQUE INDEX DC_Lockbox1 ON DC_Lockbox (DCAccountID, LockboxNumber);
CREATE SEQUENCE DCLockboxID_seq start with 1 increment by 1 nomaxvalue nocycle;
-- =====
-- This table stores summary information about lockboxes.
-- =====
CREATE TABLE DC_LockboxSummary (
  DCAccountID INTEGER NOT NULL,
  DataDate TIMESTAMP NOT NULL,
  DataSource INTEGER NOT NULL,
  TotalCredits NUMERIC(31,3),
  TotalDebits NUMERIC(31,3),
  TotalNumCredits INTEGER,
  TotalNumDebits INTEGER,
  ImmediateFloat NUMERIC(31,3),

```

```

OneDayFloat NUMERIC(31,3),
TwoDayFloat NUMERIC(31,3),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_LockboxSummary ADD PRIMARY KEY (DCAccountID, DataDate, DataClassification);
ALTER TABLE DC_LockboxSummary ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON
DELETE CASCADE;
-- =====
-- This table stores information about debits and credits to a lockbox.
-- =====
CREATE TABLE DC_LBTransactions (
DCAccountID INTEGER NOT NULL,
DCTransactionIndex BIGINT NOT NULL,
LockboxNumber VARCHAR(40) NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
TransID INTEGER,
TransTypeID INTEGER,
Description VARCHAR(1024),
Amount NUMERIC(31,3),
NumRejectedChecks INTEGER,
RejectedAmount NUMERIC(31,3),
ImmedAvailAmount NUMERIC(31,3),
OneDayAvailAmount NUMERIC(31,3),
MoreOneDayAvailAm NUMERIC(31,3),
ValueDateTime TIMESTAMP,
BankRefNum VARCHAR(40),
CustRefNum VARCHAR(40),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_LBTransactions ADD PRIMARY KEY (DCAccountID, DCTransactionIndex, DataClassification);
ALTER TABLE DC_LBTransactions ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;
-- =====
-- This table stores information about individual credit items (checks)
-- deposited in the lockbox. This table can get quite large.
-- =====
CREATE TABLE DC_LBCreditItems (
DCLockboxID INTEGER NOT NULL,
DCCreditItemIndex BIGINT NOT NULL,

```

```

DataDate      TIMESTAMP NOT NULL,
DataSource    INTEGER NOT NULL,
ItemID        INTEGER,
DocumentType  VARCHAR(40),
Payor         VARCHAR(80),
Amount        NUMERIC(31,3),
CheckNumber   VARCHAR(40),
CheckDate     DATE,
CouponAccountNum VARCHAR(40),
CouponAmount1 NUMERIC(31,3),
CouponAmount2 NUMERIC(31,3),
CouponDate1   DATE,
CouponDate2   DATE,
CouponRefNum  VARCHAR(40),
CheckRoutingNum VARCHAR(40),
CheckAccountNum VARCHAR(40),
LockboxWorkType VARCHAR(40),
LockboxBatchNum VARCHAR(40),
LockboxSeqNum VARCHAR(40),
Memo          VARCHAR(1024),
ImmedAvailAmount NUMERIC(31,3),
OneDayAvailAmount NUMERIC(31,3),
MoreOneDayAvailAm NUMERIC(31,3),
ValueDateTime TIMESTAMP,
BankRefNum    VARCHAR(40),
CustRefNum    VARCHAR(40),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra         VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_LBCreditItems ADD PRIMARY KEY (DCLockboxID, DCCreditItemIndex, DataClassification);
ALTER TABLE DC_LBCreditItems ADD FOREIGN KEY (DCLockboxID) REFERENCES DC_Lockbox ON DELETE CASCADE;

-- =====
-- Disbursement Support
-- =====
-- =====
-- This table stores summary information about disbursements.
-- =====

CREATE TABLE DC_DsbSummary (
DCAccountID  INTEGER NOT NULL,
DataDate     TIMESTAMP NOT NULL,
DataSource   INTEGER NOT NULL,
NumItemsPending INTEGER,
TotalDebits  NUMERIC(31,3),
TotalCredits NUMERIC(31,3),
TotalDTCCredits NUMERIC(31,3),

```

```

ImmedFundsNeeded NUMERIC(31,3),
OneDayFundsNeeded NUMERIC(31,3),
TwoDayFundsNeeded NUMERIC(31,3),
ValueDateTime TIMESTAMP,
ChecksPaidEarly NUMERIC(31,3),
ChecksPaidLate NUMERIC(31,3),
ChecksPaidLast NUMERIC(31,3),
FedEstimate NUMERIC(31,3),
LateDebits NUMERIC(31,3),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_DsbSummary ADD PRIMARY KEY (DCAccountID, DataDate, DataClassification);
ALTER TABLE DC_DsbSummary ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;
-- =====
-- This table stores information about individual disbursements.
-- This table can get quite large.
-- =====

CREATE TABLE DC_DsbTransactions (
DCAccountID INTEGER NOT NULL,
DCTransactionIndex BIGINT NOT NULL,
DataDate TIMESTAMP NOT NULL,
DataSource INTEGER NOT NULL,
TransID INTEGER,
CheckDate DATE,
Payee VARCHAR(40),
Amount NUMERIC(31,3),
CheckNumber VARCHAR(40),
CheckRefNum VARCHAR(40),
Memo VARCHAR(1024),
IssuedBy VARCHAR(80),
ApprovedBy VARCHAR(80),
ImmedFundsNeeded NUMERIC(31,3),
OneDayFundsNeeded NUMERIC(31,3),
TwoDayFundsNeeded NUMERIC(31,3),
ValueDateTime TIMESTAMP,
BankRefNum VARCHAR(40),
CustRefNum VARCHAR(40),
Presentment VARCHAR(40),
BAIFileIdentifier VARCHAR(255),
ExtendABeanXMLID BIGINT NOT NULL,
Extra VARCHAR(1024),
DataSourceFileName VARCHAR(255),
DataSourceFileDate TIMESTAMP,
DataClassification CHAR(1) NOT NULL

```



```

);
ALTER TABLE DC_DsbTransactions ADD PRIMARY KEY (DCAccountID, DCTransactionIndex, DataClassification);
ALTER TABLE DC_DsbTransactions ADD FOREIGN KEY (DCAccountID) REFERENCES DC_Account ON DELETE
CASCADE;
-- =====
-- This table stores the ExtendABean xml
-- DCExtendABeanID is generated from the DCExtendABean row in the
-- DC_RecordCounter table
-- =====
CREATE TABLE DC_ExtendABeanXML (
    DCExtendABeanID INTEGER NOT NULL,
    XMLSegmentNumber INTEGER NOT NULL,
    XMLSegment VARCHAR(2000)
);
ALTER TABLE DC_ExtendABeanXML ADD PRIMARY KEY (DCExtendABeanID, XMLSegmentNumber);
-- =====
-- This table stores information about which index to use for a newly
-- inserted record, dependent on which type of record it is. This table
-- is used to support paging APIs.
--
-- There are counters per account/lockbox.
--
-- Defined object types are:
-- 1 = accountID
-- 2 = lockboxID
-- 3 = extendABeanInfo
--
-- If an account or lockbox is removed, then this table needs to be cleaned up.
--
-- Valid counter names are:
-- DsbTrans
-- AccTrans
-- LBTrans
-- LBCreditItem
-- DCExtendABeanID
-- =====
CREATE TABLE DC_RecordCounter (
    DCOBJECTType INTEGER NOT NULL,
    DCOBJECTID INTEGER NOT NULL,
    CounterName VARCHAR(40) NOT NULL,
    NextIndex BIGINT NOT NULL,
    DataClassification CHAR(1) NOT NULL
);
ALTER TABLE DC_RecordCounter ADD PRIMARY KEY (DCOBJECTType, DCOBJECTID, CounterName,
DataClassification);
INSERT INTO DC_RecordCounter ( DCOBJECTType, DCOBJECTID, CounterName, NextIndex, DataClassification )
VALUES ( 3, 0, 'DCExtendABeanIndex', 1, 'P' );

```